

创建Maven SpringMVC项目

创建Maven SpringMVC项目

一、创建Maven Web项目

二、Maven自动导入jar包

三、创建Java目录

四、Spring MVC框架配置

1. web.xml配置

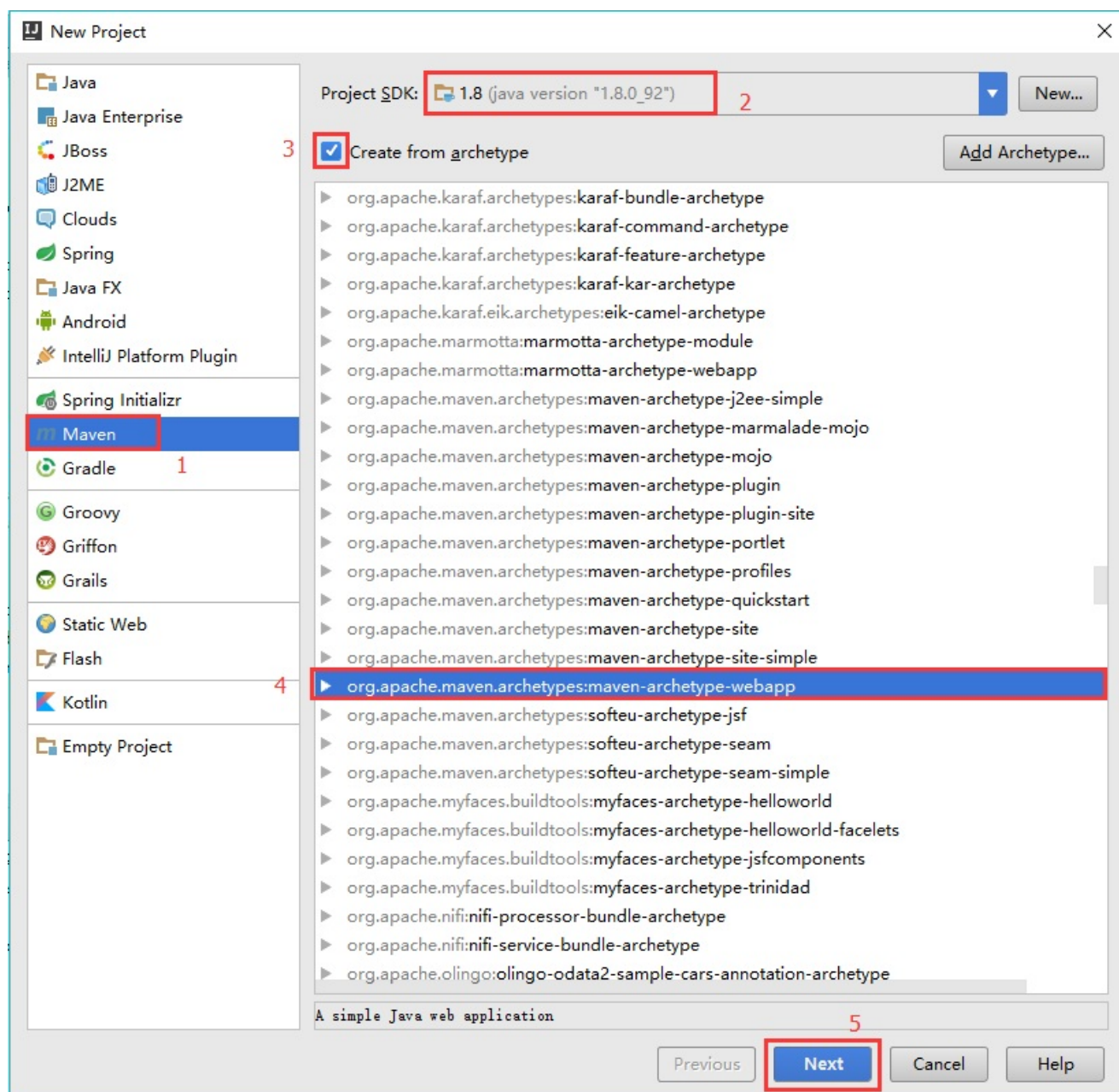
2. xxx-servlet.xml配置

五、运行

六、问题

1. no declaration can be found for element 'context:component-scan'

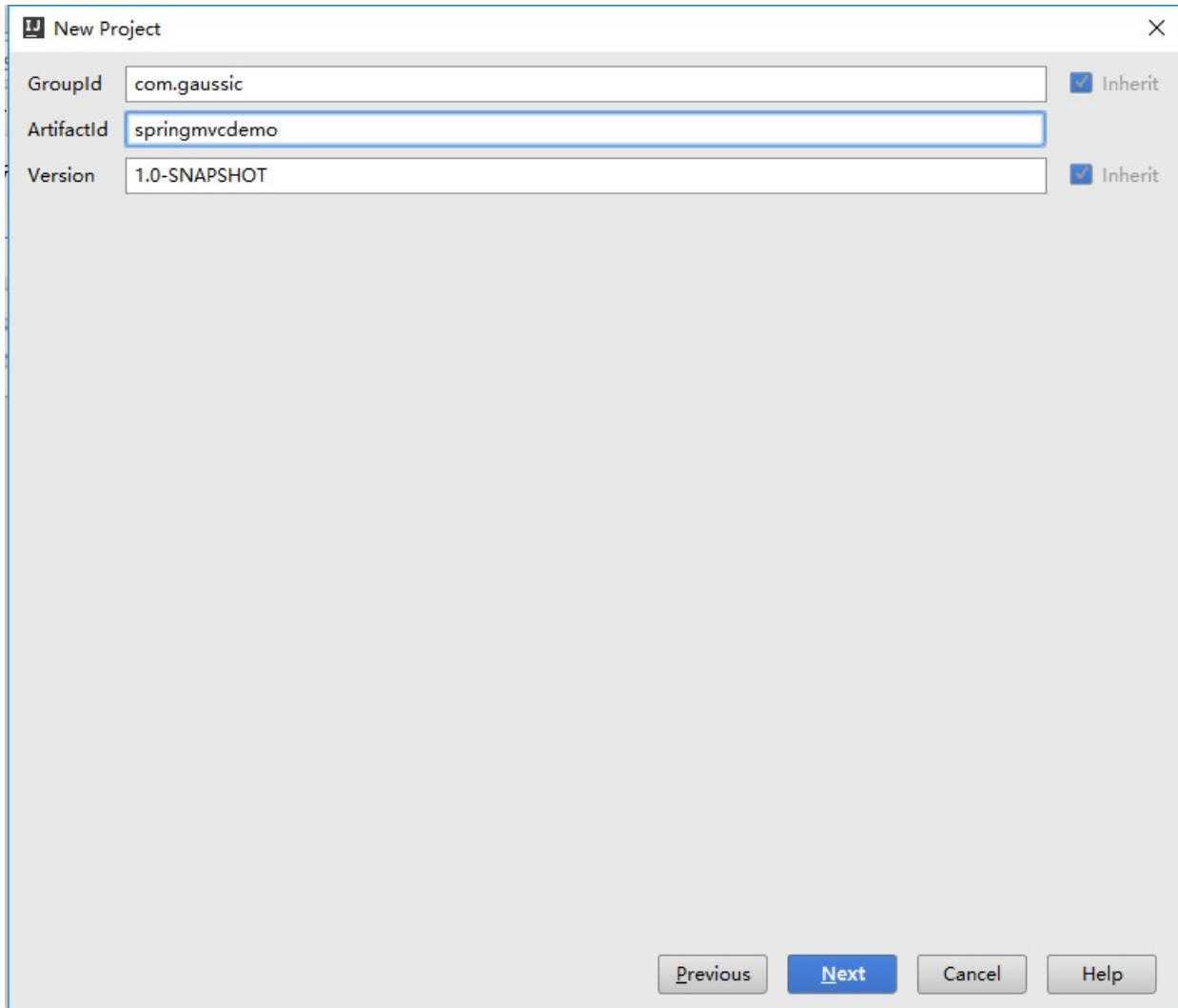
一、创建Maven Web项目



1. 选择左边Maven
2. 设置Project SDK
3. 勾选Create from archetype

4. 选择`org.apache.maven.archetypes:maven-archetype-webapp`

5. 点击**Next**



New Project

GroupId: com.gaussic ☒ Inherit

ArtifactId: springmvcdemo ☒ Inherit

Version: 1.0-SNAPSHOT ☒ Inherit

Previous Next Cancel Help

6. 填入**GroupId**、**ArtifactId**、**Version**，这三个属性标识了项目的唯一性，例如Tomcat的GroupId是org.apache，即它是apache组织的项目，ArtifactId是tomcat，项目名为tomcat。一般只有在发布时需要保证这三者的唯一性；

7. 点击**Next**

New Project

Maven home directory: Bundled (Maven 3) (Version: 3.0.5)

User settings file: C:\Users\dzkan\.m2\settings.xml ☐ Override

Local repository: C:\Users\dzkan\.m2\repository ☐ Override

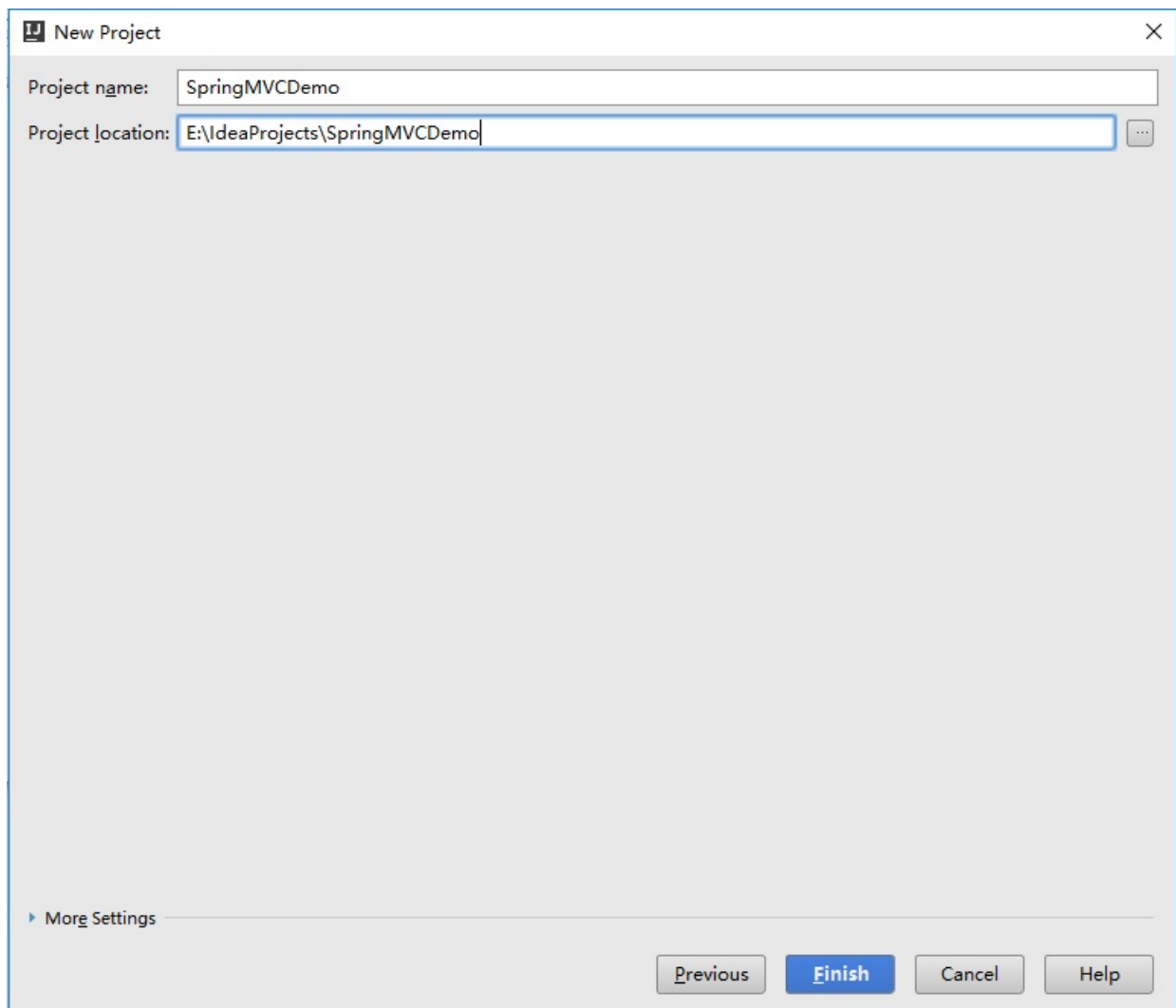
Properties

groupId	com.gaussian
artifactId	springmvcdemo
version	1.0-SNAPSHOT
archetypeGroupId	org.apache.maven.archetypes
archetypeArtifactId	maven-archetype-webapp
archetypeVersion	RELEASE

Previous Next Cancel Help

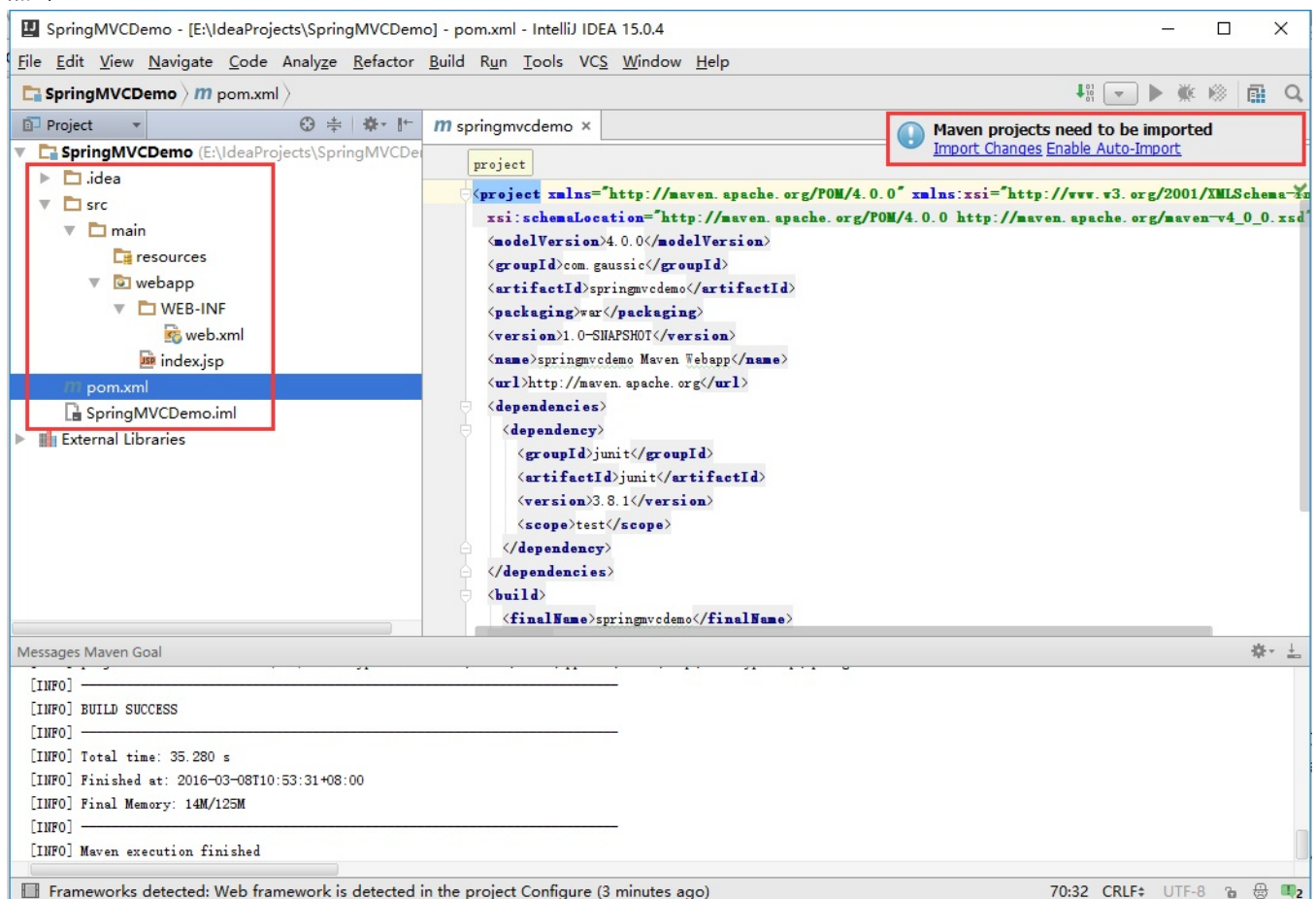
8. 设置Maven home direction

9. 点击Next



10. 填写项目的名字已经保存路径：Project name、Project location

11. 点击Finish



12. 项目生成的结构图如上图：

- `src/main/resources` 文件夹：一般用来存放一些资源文件
 - `src/main/webapp` 文件夹：用来存放web配置文件以及jsp页面等（加上resource已经构成一个web应用）
 - 右上角的 `Enable-Auto-Import` ：可以在每次修改pom.xml后，自动的下载并导入jar包
-

二、Maven自动导入jar包

Maven通过pom文件将项目所需要的jar包从几个中央仓库（其中最常用的是[Maven Repository](https://maven.apache.org)）下载到本地repo目录下，然后关联到项目中来。打开根目录的 `pom.xml` 文件：

```
1. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2.     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.
   xsd">
3.     <modelVersion>4.0.0</modelVersion>
4.     <groupId>com.gaussic</groupId>
5.     <artifactId>springmvcdemo</artifactId>
6.     <packaging>war</packaging>
7.     <version>1.0-SNAPSHOT</version>
8.     <name>springmvcdemo Maven Webapp</name>
9.     <url>http://maven.apache.org</url>
10.    <dependencies>
11.        <dependency>
12.            <groupId>junit</groupId>
13.            <artifactId>junit</artifactId>
14.            <version>3.8.1</version>
15.            <scope>test</scope>
16.        </dependency>
17.    </dependencies>
18.    <build>
19.        <finalName>springmvcdemo</finalName>
20.    </build>
21. </project>
```

该文件包含了本项目的groupId等信息，而 `<dependencies>` 标签，翻译过来是“依赖”的意思，也就是说把对每个包的需求都称为一个依赖 `<dependency>`，定义在 `<dependencies>` 中。在每个 `<dependency>` 中，**需要提供的是所需jar包的groupId、artifactId、version这三个必要信息**，比如引入一个junit包，格式如下：

```
1. <dependency>
2.     <groupId>junit</groupId>
3.     <artifactId>junit</artifactId>
4.     <version>3.8.1</version>
5.     <scope>test</scope>
6. </dependency>
```

可以到[Maven Repository](https://maven.apache.org)中搜索自己所需要的jar包，然后复制底下的Maven依赖，复制到项目中来（但一般IDEA能够根据填写的artifactId自动填写对应的groupId以及version，比较方便）

Spring Core » 4.2.5.RELEASE



Spring Core

2,147 usages

org.springframework » spring-core » 4.2.5.RELEASE under **Core Utilities**

Spring Core

Artifact	Download (JAR) (1.1 MB)
POM File	View
Date	(Feb 25, 2016)
HomePage	https://github.com/spring-projects/spring-framework
Organization	Spring IO
Issue Tracker	https://jira.springsource.org/browse/SPR

Maven

Ivy

Grape

Gradle

Buildr

SBT

Leiningen

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>4.2.5.RELEASE</version>
</dependency>
```

添加其他依赖，以及修改 `<build>`，最后pom文件如下：

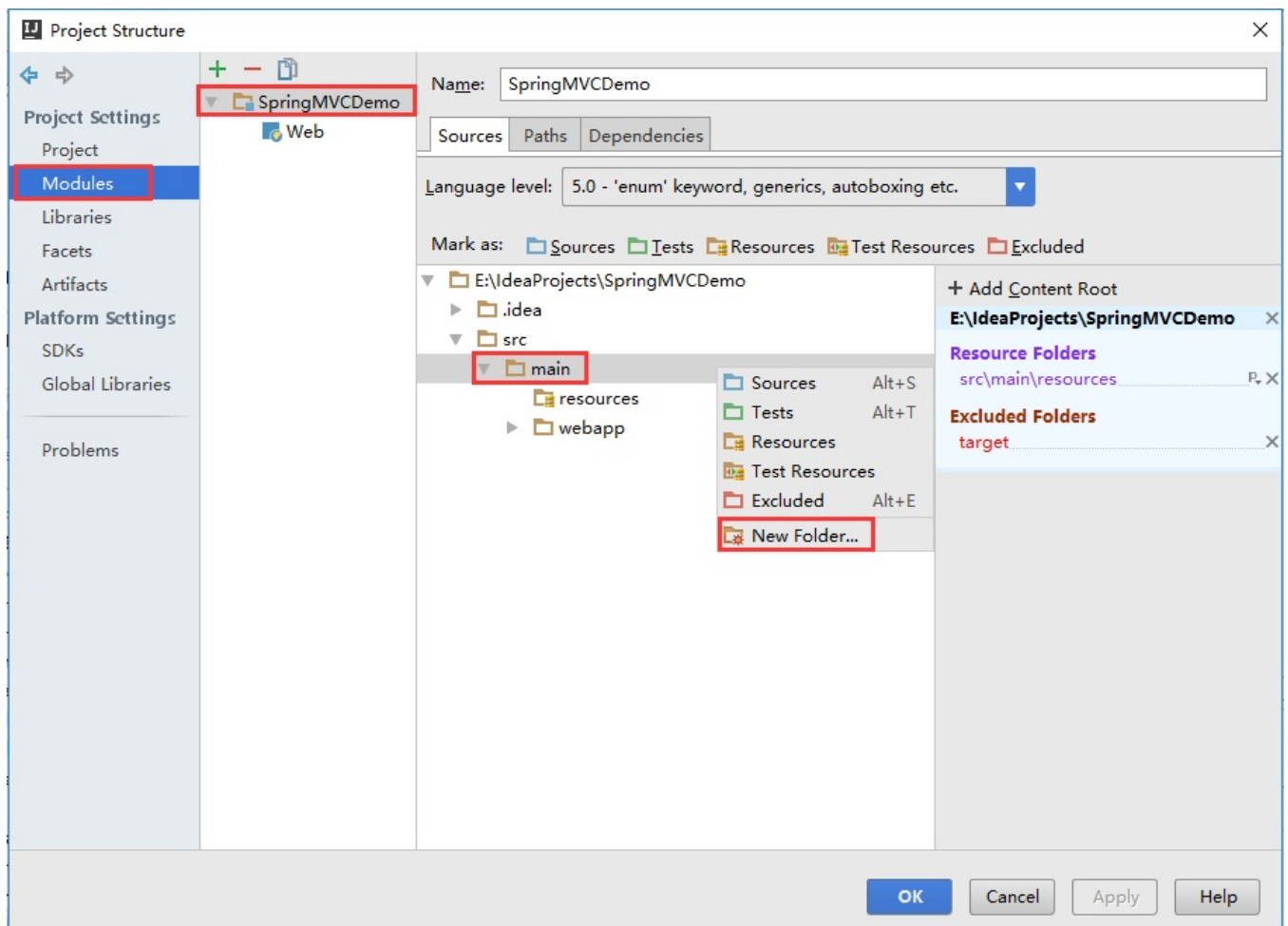
```

1.
2. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3.         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.
   xsd">
4.     <modelVersion>4.0.0</modelVersion>
5.     <groupId>com.pentb</groupId>
6.     <artifactId>springmvcdemo</artifactId>
7.     <packaging>war</packaging>
8.     <version>1.0-SNAPSHOT</version>
9.     <name>springmvcdemo Maven Webapp</name>
10.    <url>http://maven.apache.org</url>
11.
12.    <properties>
13.        <spring.version>4.2.2.RELEASE</spring.version>
14.
15.    </properties>
16.
17.    <dependencies>
18.        <dependency>
19.            <groupId>junit</groupId>
20.            <artifactId>junit</artifactId>
21.            <version>3.8.1</version>
22.            <scope>test</scope>
23.        </dependency>
24.
25.        <dependency>
26.            <groupId>org.springframework</groupId>
27.            <artifactId>spring-core</artifactId>
28.            <version>${spring.version}</version>
29.        </dependency>
30.
31.        <dependency>
32.            <groupId>org.springframework</groupId>
33.            <artifactId>spring-context</artifactId>
34.            <version>4.2.2.RELEASE</version>
35.        </dependency>
36.
37.        <dependency>
38.            <groupId>org.springframework</groupId>
39.            <artifactId>spring-webmvc</artifactId>
40.            <version>${spring.version}</version>
41.        </dependency>
42.
43.        <dependency>
44.            <groupId>javax.servlet</groupId>
45.            <artifactId>jstl</artifactId>
46.            <version>1.2</version>
47.        </dependency>
48.
49.    </dependencies>
50.    <build>
51.        <finalName>springmvcdemo</finalName>
52.        <plugins>
53.            <plugin>
54.                <groupId>org.apache.maven.plugins</groupId>
55.                <artifactId>maven-compiler-plugin</artifactId>
56.                <configuration>
57.                    <source>1.8</source>
58.                    <target>1.8</target>
59.                </configuration>
60.            </plugin>
61.        </plugins>
62.    </build>
63. </project>

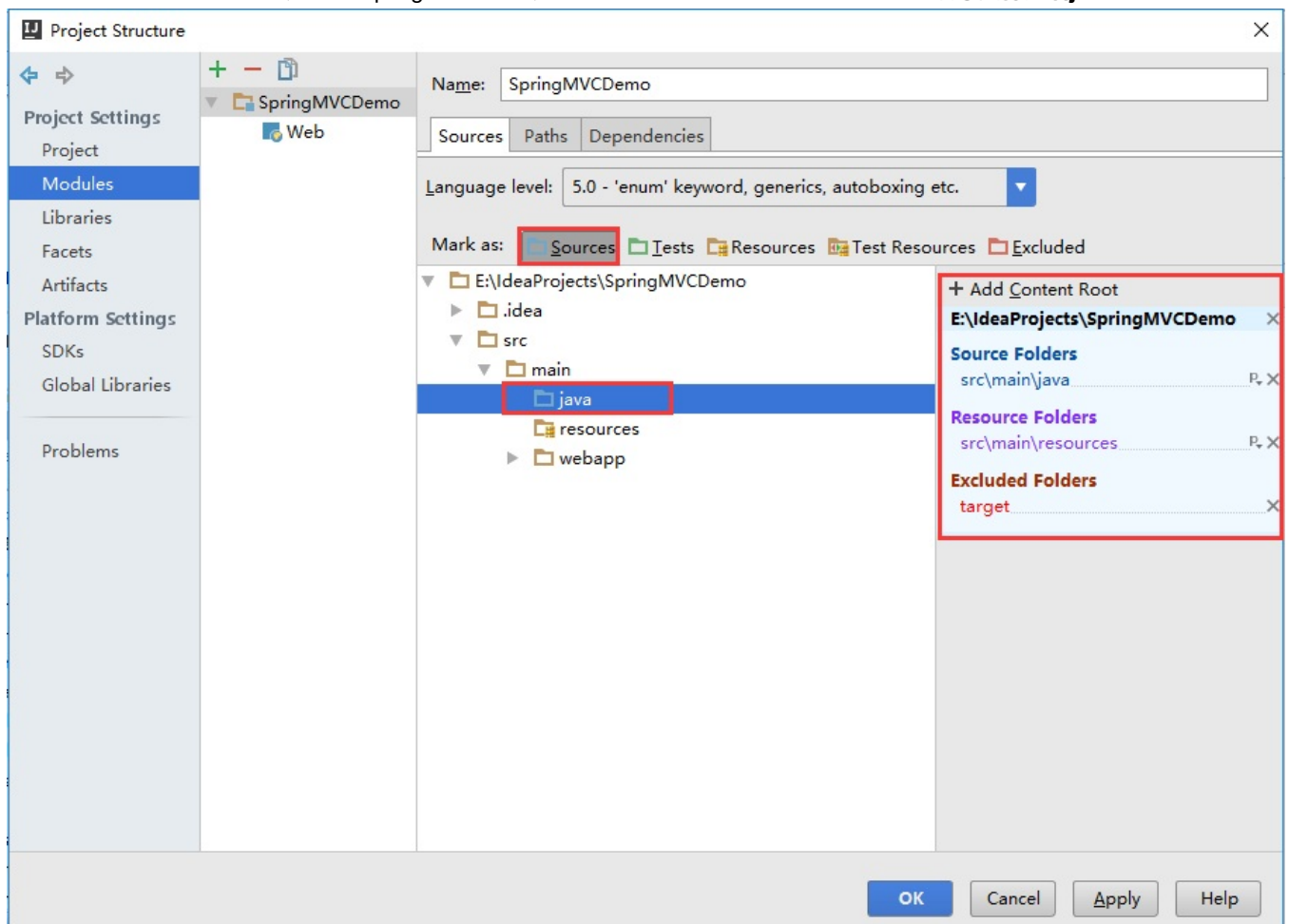
```

三、创建Java目录

1. File -> Project Structure



2. **Modules** -> 创建的项目名（如上图SpringMVCDemo） -> **Source** -> **main** -> **New Folder** -> 填写文件夹名java



3. 将Java目录标记为Sources

四、Spring MVC框架配置

1. web.xml配置

1. 打开 `src\main\webapp\WEB-INF\web.xml` 文件，稍微更新一下web.xml的版本，可以支持更高级的一些语法，如下：

```
1.
2. <?xml version="1.0" encoding="UTF-8"?>
3. <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
4.     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5.     xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/
web-app_3_1.xsd"
6.     version="3.1">
7.
8.     <display-name>SpringMVCDemo Web Application</display-name>
9.
10. </web-app>
```

2. 在 `<web-app>` 中加入一个 `<servlet>`（该类为Spring MVC框架中的类）：

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3.     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4.     xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/
web-app_3_1.xsd"
5.     version="3.1">
6.
7.     <display-name>SpringMVCDemo Web Application</display-name>
8.
9.     <servlet>
10.         <servlet-name>mvc-dispatcher</servlet-name> <!-- 名字可自定义 -->
11.         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class> <!-- 使用Sp
ring MVC框架的类 -->
12.         <load-on-startup>1</load-on-startup>
13.     </servlet>
14.
15.     <servlet-mapping> <!-- Servlet必须有对应的mapping—匹配对应的url -->
16.         <servlet-name>mvc-dispatcher</servlet-name>
17.         <url-pattern>/</url-pattern>
18.     </servlet-mapping>
19. </web-app>
```

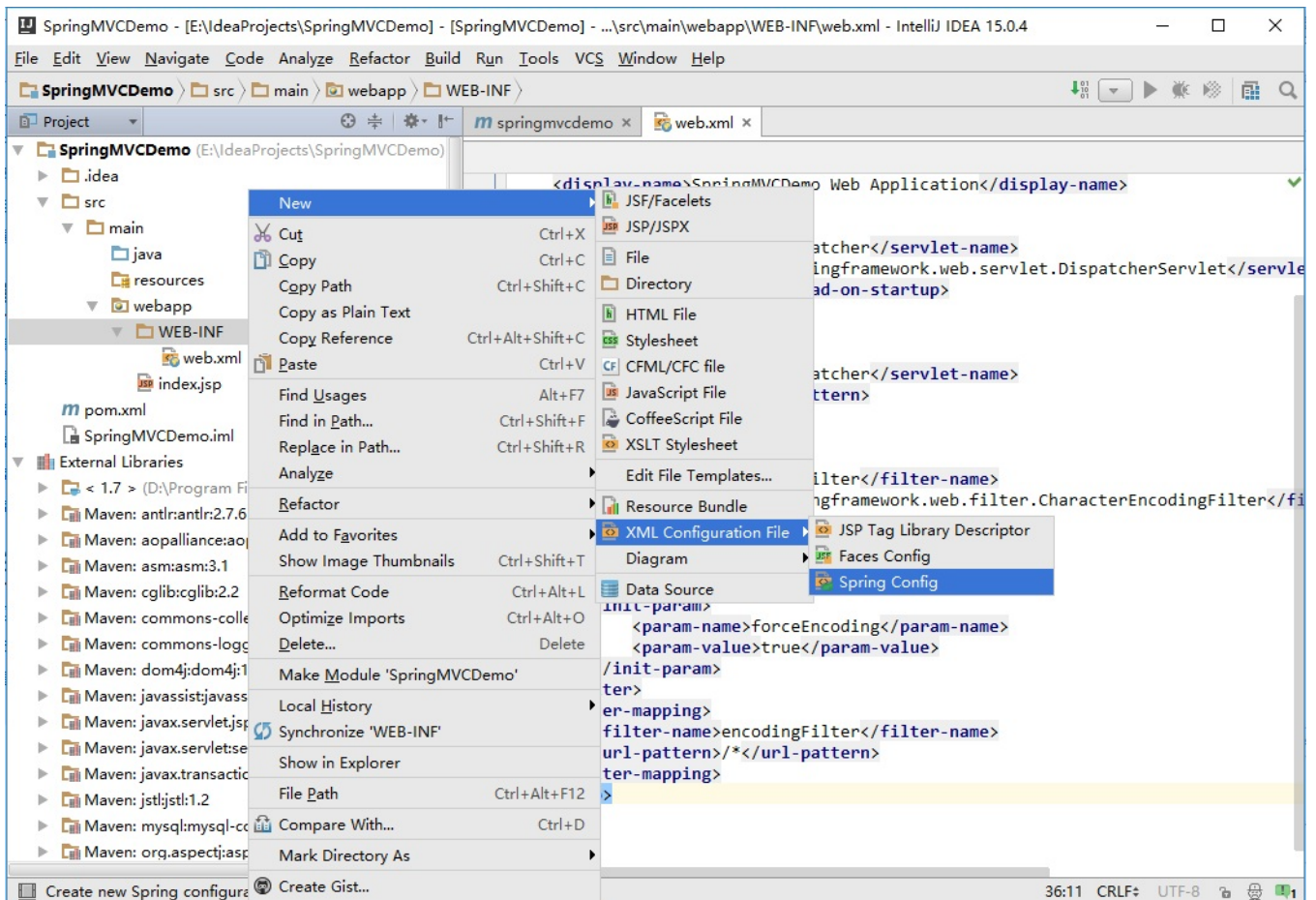
该servlet名为mvc-dispatcher（名称可修改），用于拦截请求（url-pattern为/，说明拦截所有请求），并交由Spring MVC的后台控制器来处理。这一项配置是必须的。

3. 为了能够处理中文的post请求，配置一个encodingFilter（），避免post请求中文时出现乱码的情况：

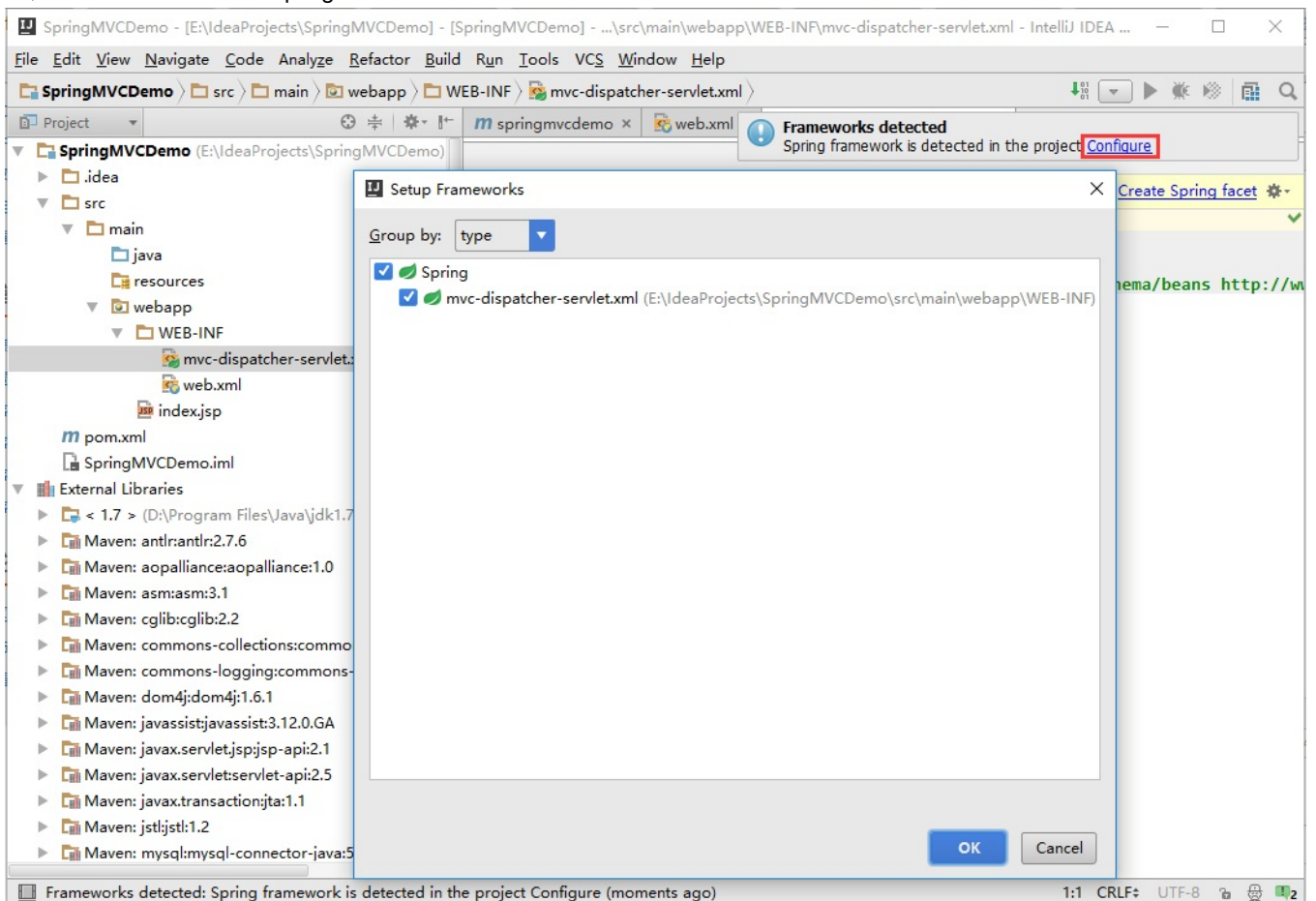
```
1. <filter>
2. <filter-name>encodingFilter</filter-name> <!-- 名字可自定义 -->
3. <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class> <!-- 使用Spr
ing MVC框架的类 -->
4. <init-param>
5.     <param-name>encoding</param-name>
6.     <param-value>UTF-8</param-value>
7. </init-param>
8. <init-param>
9.     <param-name>forceEncoding</param-name>
10.    <param-value>true</param-value>
11. </init-param>
12. </filter>
13. <filter-mapping>
14. <filter-name>encodingFilter</filter-name>
15. <url-pattern>/*</url-pattern>
16. </filter-mapping>
```

2. xxx-servlet.xml配置

1. 新建 `xxx-servlet.xml`：在配置完web.xml后，需在web.xml同级目录下新建 `mvc-dispatcher-servlet.xml`（注：`-servlet` 前面的 `mvc-dispatcher` 是在web.xml中配置的servlet名字）：



2. 设置Setup Frameworks：新建该xml文件后，点击右上角的configure，出现 Setup Frameworks 界面，点击OK，这样，IntelliJ IDEA就识别了SpringMVC的配置文件：



3. mvc-dispatcher-servlet文件
mvc-dispatcher-servlet.xml文件如下：

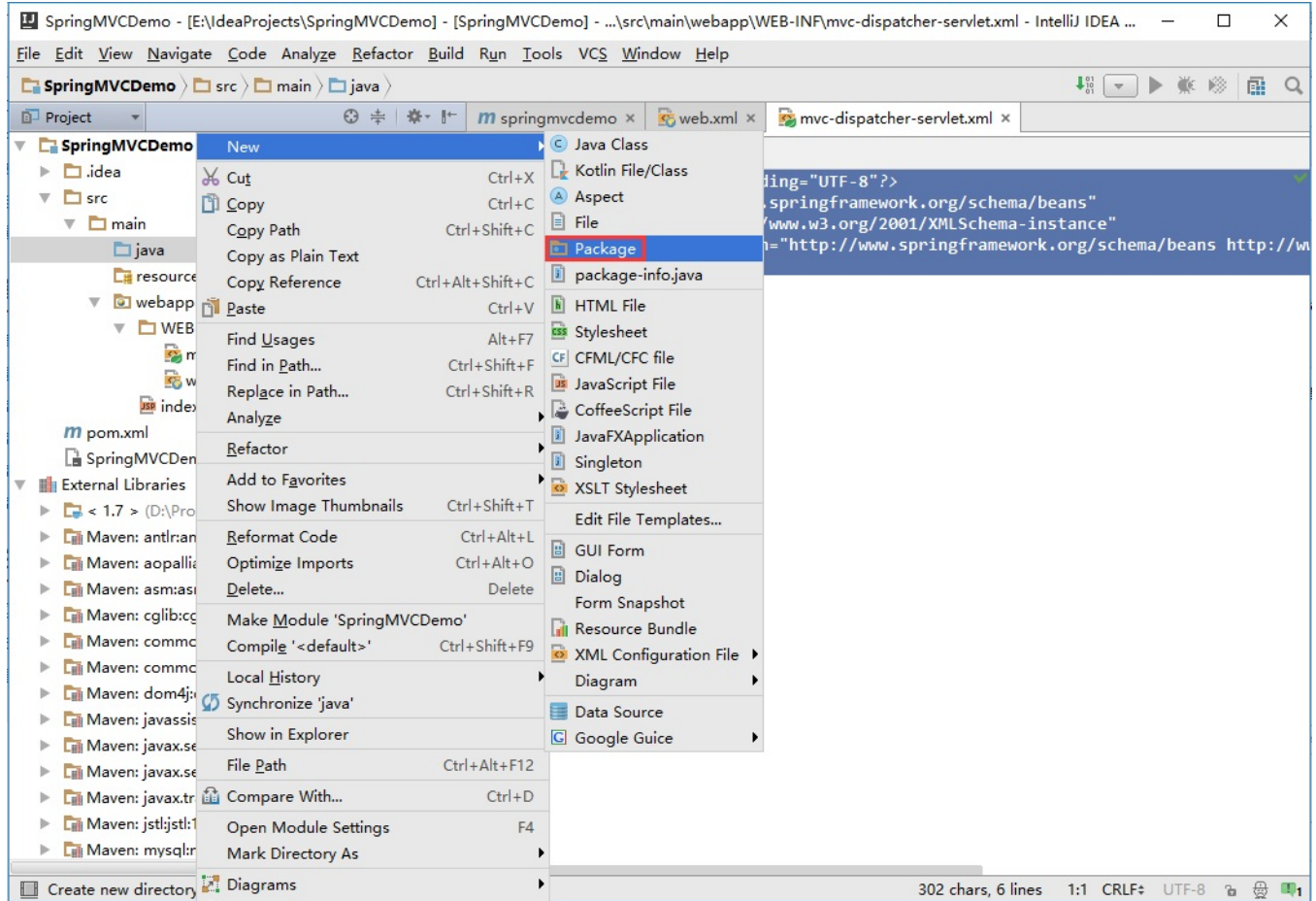
```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3.       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4.       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
5.
6. </beans>

```

4. 编写Controller

MVC框架有model、view、controller三部分组成。model一般为一些基本的Java Bean，view用于进行相应的页面显示，controller用于处理网站的请求。在 `src\main\java` 中新建一个用于保存controller的package：



在controller包中新建java类MainController（名称并不固定，可任意取），并修改如下：

```

1. package com.gaussic.controller;
2.
3. import org.springframework.stereotype.Controller;
4. import org.springframework.web.bind.annotation.RequestMapping;
5. import org.springframework.web.bind.annotation.RequestMethod;
6.
7. /**
8.  * Created by dzkan on 2016/3/8.
9.  */
10. @Controller
11. public class MainController {
12.
13.     @RequestMapping(value = "/", method = RequestMethod.GET)
14.     public String index() {
15.         return "index";
16.     }
17. }

```

- `@Controller` 注解：采用注解的方式，可以明确地定义该类为处理请求的Controller类；
- `@RequestMapping()` 注解：用于定义一个请求映射，value为请求的url，值为 / 说明，该请求首页请求，method用以指定该请求类型，一般为get和post；
- `return "index"`：处理完该请求后返回的页面，此请求返回 index.jsp页面。

5. 配置mvc-dispatcher-servlet.xml

- 加入component-scan标签，指明controller所在的包，并扫描其中的注解（最好不要复制，输入时按IDEA会在beans xmlns中添加相关内容）：

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3.     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4.     xmlns:context="http://www.springframework.org/schema/context"
5.     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">
6.
7.     <!--指明 controller 所在包，并扫描其中的注解-->
8.     <context:component-scan base-package="com.gaussic.controller"/>
9. </beans>
```

- 再进行js、image、css等静态资源访问的相关配置，这样，SpringMVC才能访问网站内的静态资源：

```
1. <!-- 静态资源(js、image等)的访问 -->
2. <mvc:default-servlet-handler/>
```

- 再开启springmvc注解模式，由于我们利用注解方法来进行相关定义，可以省去很多的配置：

```
1. <!-- 开启注解 -->
2. <mvc:annotation-driven/>
```

- 再进行视图解析器的相关配置，关于controller如何找到视图文件？在 controller 的一个方法中，返回的字符串定义了所需访问的jsp的名字（如上面的index）。在 `JspViewResolver` 中，有两个属性，一个是 **prefix**，定义了所需访问的文件路径前缀，另一是 **suffix**，表示要访问的文件的后缀，这里为 `.jsp`。那么，如果返回字符串是 xxx，SpringMVC就会找到 `/WEB-INF/pages/xxx.jsp` 文件。

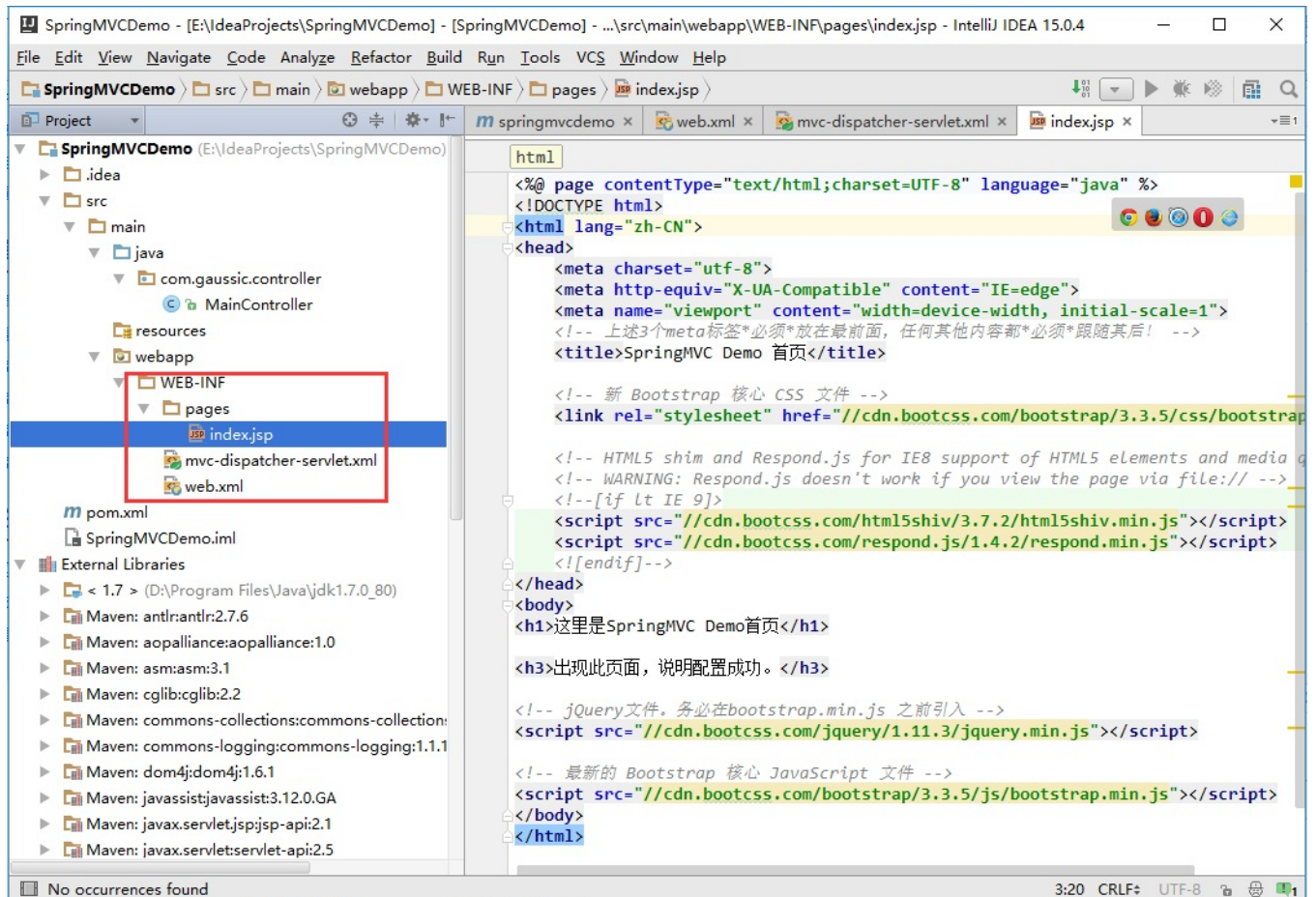
```
1. <!--ViewResolver 视图解析器-->
2. <!--用于支持Servlet、JSP视图解析-->
3. <bean id="jspViewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
4.     <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
5.     <property name="prefix" value="/WEB-INF/pages/" />
6.     <property name="suffix" value=".jsp" />
7. </bean>
```

- 完整mvc-dispatcher-servlet配置

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3.     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4.     xmlns:context="http://www.springframework.org/schema/context"
5.     xmlns:mvc="http://www.springframework.org/schema/mvc"
6.     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd">
7.
8.     <!--指明 controller 所在包，并扫描其中的注解-->
9.     <context:component-scan base-package="com.gaussic.controller"/>
10.
11.     <!-- 静态资源(js、image等)的访问 -->
12.     <mvc:default-servlet-handler/>
13.
14.     <!-- 开启注解 -->
15.     <mvc:annotation-driven/>
16.
17.     <!--ViewResolver 视图解析器-->
18.     <!--用于支持Servlet、JSP视图解析-->
19.     <bean id="jspViewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
20.         <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
21.         <property name="prefix" value="/WEB-INF/pages/" />
22.         <property name="suffix" value=".jsp" />
23.     </bean>
24. </beans>
```

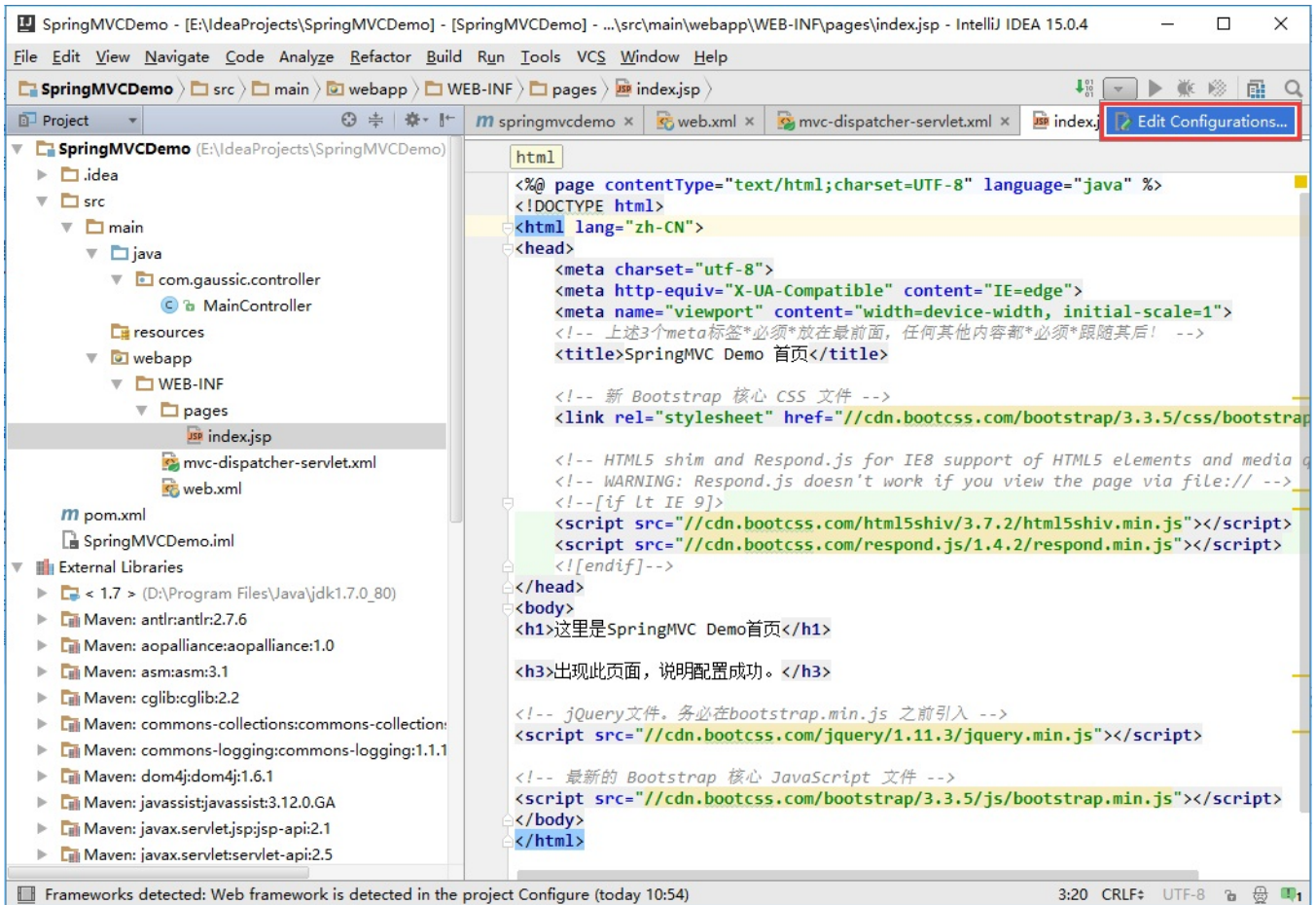

- 调整项目小部分结构：删除 `/webapp/index.jsp` 文件，在 `WEB-INF` 目录下新建文件夹 `pages`，再在 `pages` 目录下新建 `index.jsp`，并修改为如下所示：

```
1. <%@ page contentType="text/html;charset=UTF-8" language="java" %>
2. <html>
3. <head>
4. <meta charset="UTF-8">
5. <meta http-equiv="X-UA-COMPATIBLE" content="IE=edge">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. <title>Spring MVC</title>
8. </head>
9. <body>
10. <h1>Spring MVC Demo首页</h1>
11. Hello Spring MVC !
12. </body>
13. </html>
```

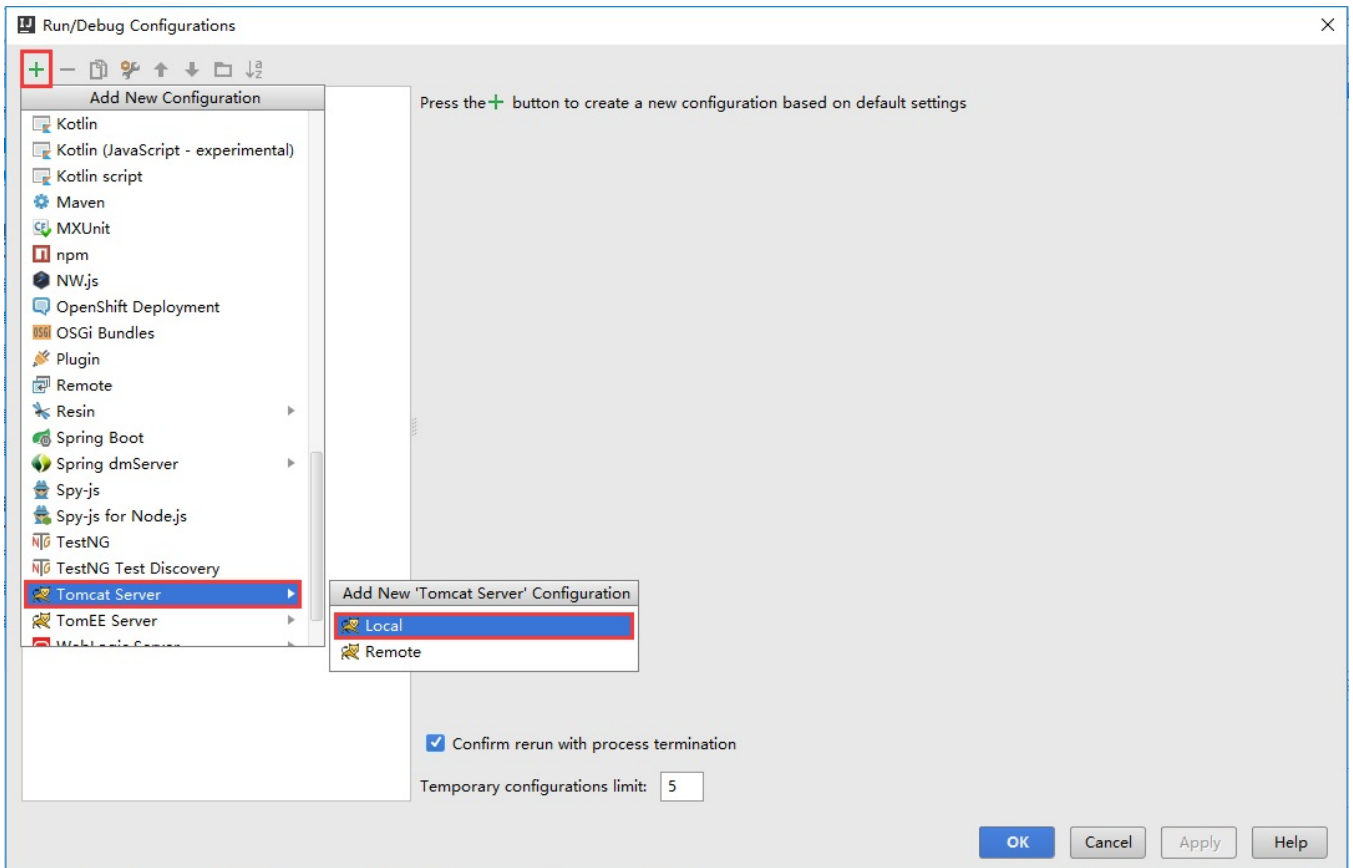


五、运行

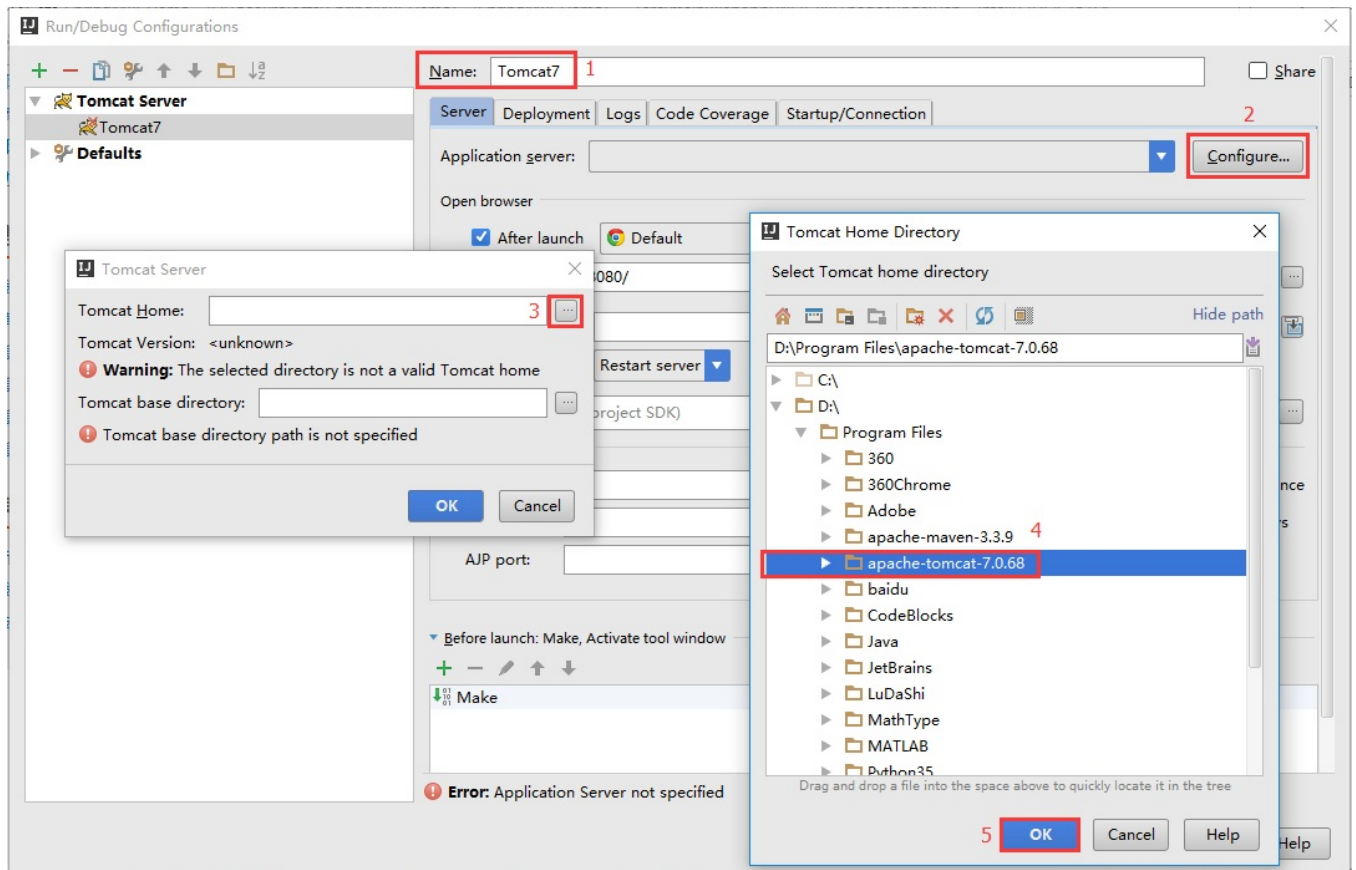
1. 点击右上角的运行配置“Edit Configurations”：



2. 点击左上角的“+”号，选择Tomcat Server，（如果没有请选择最下方的33 items more，找到Tomcat Server），再选择Local：

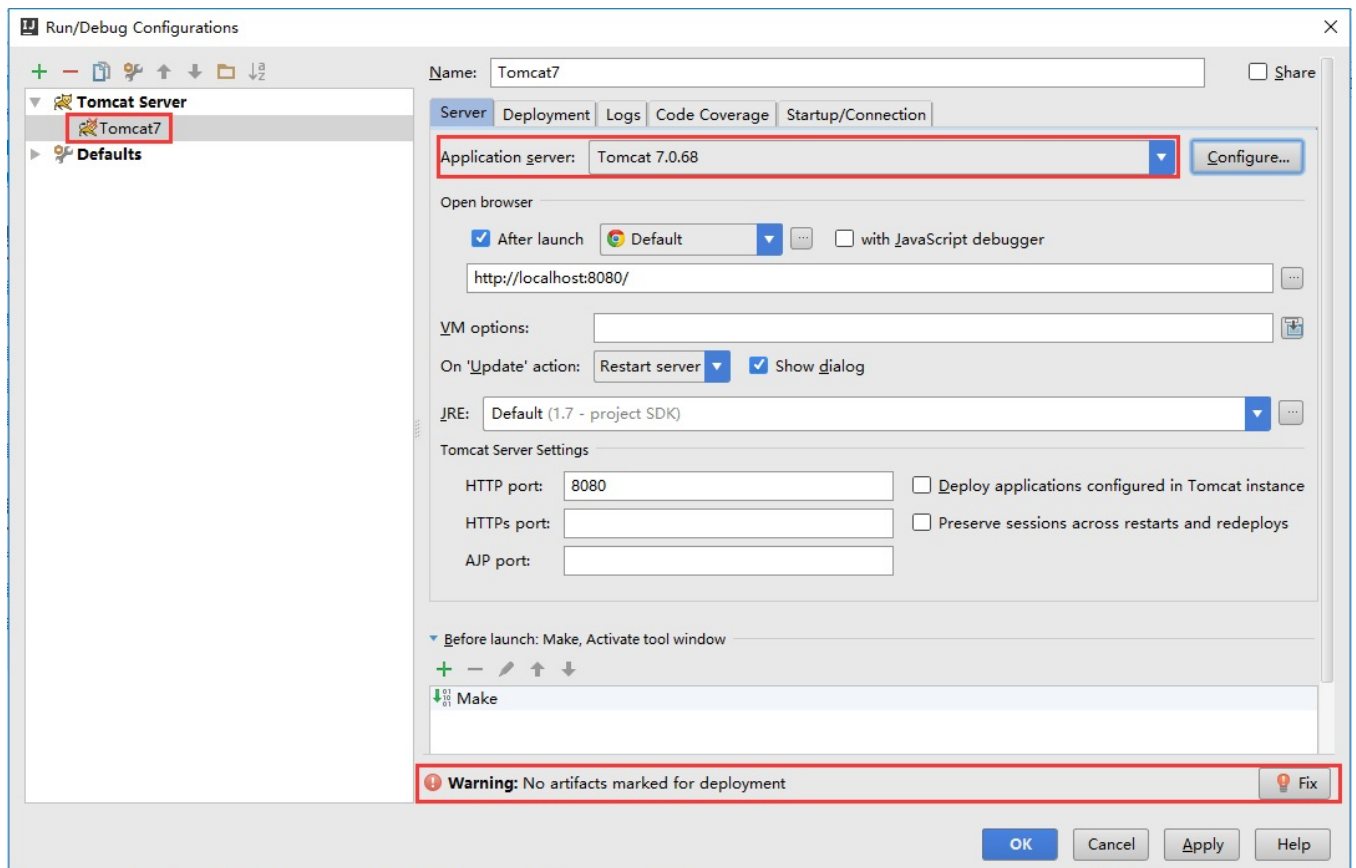


3. 配置Tomcat：

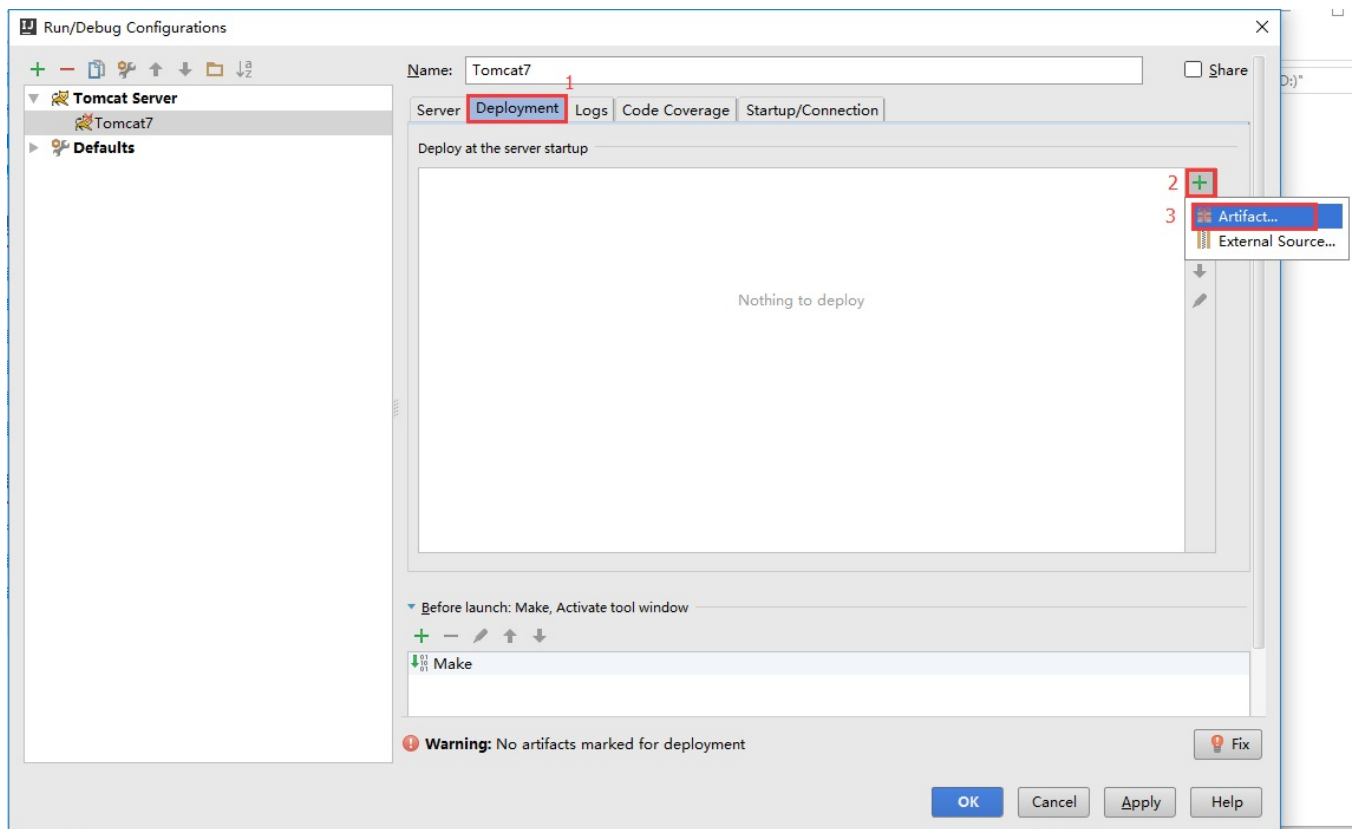


- 修改运行名称Name
- 配置服务器Configure...
- 选择Tomcat的目录Tomcat Home

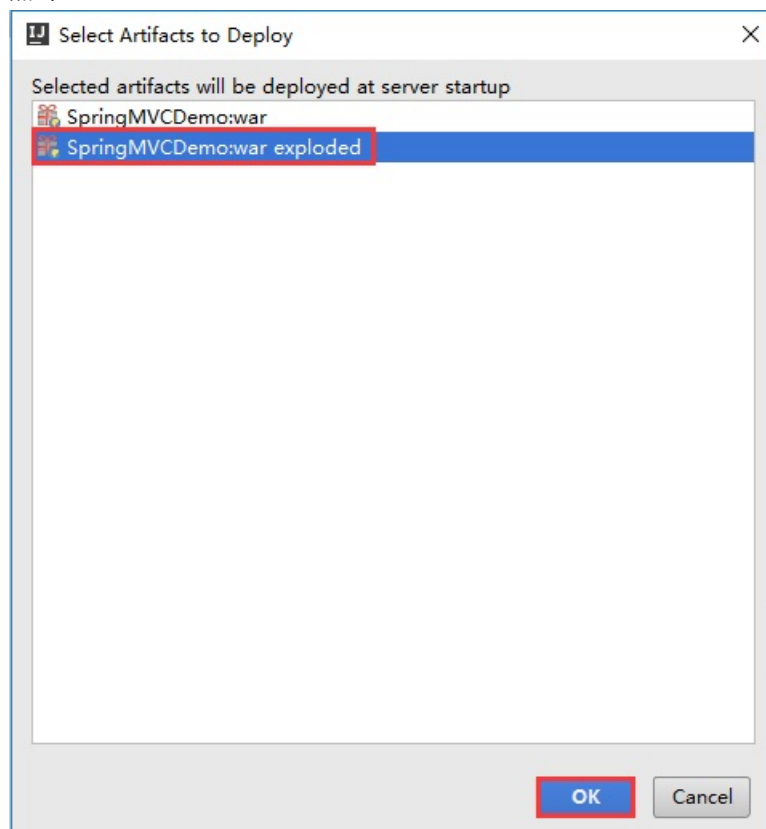
4. 部署到Tomcat



在配置好tomcat的路径后，如下图所示，发现依然存在警告，且左方的Tomcat7图标上有一个错误标记，说明还没有配置完全：



- 选择**Deployment**
- 点击“+”号
- 选择第二个：**war exploded**
- 点击**OK**



5. 配置完成，点击运行即可自动启动浏览器

六、问题

1. no declaration can be found for element 'context:component-scan'

1. 原因：没有指定 `mvc-dispatcher-servlet.xml` 文件中context名字空间的scheme位置。

2. 解决

将 `mvc-dispatcher-servlet.xml` 文件的 `themeLocation` 修改为：

```
1.  xsi:schemaLocation="
2.  http://www.springframework.org/schema/beans
3.  http://www.springframework.org/schema/beans/spring-beans.xsd
4.  http://www.springframework.org/schema/context
5.  http://www.springframework.org/schema/context/spring-context.xsd
6.  http://www.springframework.org/schema/mvc
7.  http://www.springframework.org/schema/mvc/spring-mvc.xsd"
```

其

中 `http://www.springframework.org/schema/context` 和 `http://www.springframework.org/schema/context/spring-context.xsd` 为缺少的内容。